A Appendix

A.1 Cases breaking periodic invariance

We show two graph construction methods that break periodic invariance in Fig. 1 and Fig. 2.

One is the graph construction method employed by Graphormer in OC20 [1], shown in Fig. 1. It breaks periodic invariance because it treats every atom as a single node in the constructed graph. When the periodic boundaries are shifted, the inner structure can change a lot in a unit cell. Hence, the constructed fully connected graphs can be totally different when periodic boundaries are shifted.

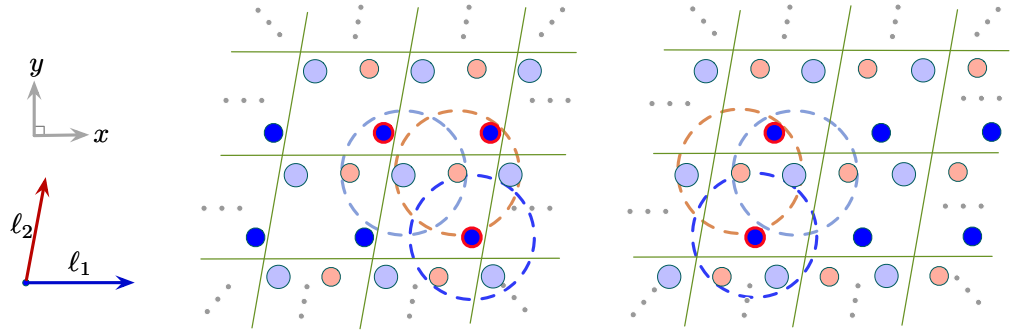

Figure 1: Illustration of graph construction method used by Graphormer [11] in OC20 [1]. We use orange, blue, and light blue to mark different atoms. We use circles of the same colors as atoms to denote the corresponding radius for atoms. It can be seen from the comparison of the left and the right that the constructed graphs are totally different when the periodic boundaries are shifted. In particular, the constructed graph on the left has three blue atoms, but the constructed graph on the right has two blue atoms. Thus, the graph construction method employed by Graphormer in OC20 breaks periodic invariance for crystals.

Another is the graph construction method using nearest neighbors based only on geometric pairwise distances, employed by CGCNN [10] and GATGNN [7], shown in Fig. 2. If several different atoms have the same geometric distances to the center atom, there is no deterministic way to determine which one to choose. Thus, for these cases, this method breaks periodic invariance.

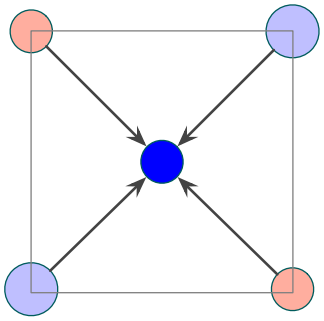

Figure 2: An illustration that crystal graph construction using nearest neighbors based only on pairwise distances breaks periodic invariance. We illustrate the one nearest neighbor case for simplicity. Black arrows denote the same pairwise distances from nearby atoms to the center blue atom. If sorting is based only on distances, there will be two different neighborhood formation approaches for the center blue atom, either using the light-blue atom or using the orange atom as the first nearest neighbor.

A.2 Proofs of periodic invariance

Notations. Recall that we use matrices $\mathbf{A}$, $\mathbf{P}$, and $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3]^T \in \mathbb{R}^{3\times 3}$ to describe a given crystal structure. $\mathbf{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n]^T \in \mathbb{R}^{n\times d_a}$ is the atom feature matrix, where $\boldsymbol{a}_i \in \mathbb{R}^{d_a}$ is the d_a-dimensional feature vector for atom i in the unit cell. $\mathbf{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n]^T \in \mathbb{R}^{n\times 3}$ is the position matrix, where $\boldsymbol{p}_i \in \mathbb{R}^3$ contains the Cartesian coordinates for atom i in 3D space. And the

infinite crystal structure can be represented as $\hat{\mathbf{P}} = \{\hat{\boldsymbol{p}}_i | \hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}, i \in \mathbb{Z}, 1 \leq i \leq n\}$, and $\hat{\mathbf{A}} = \{\hat{\boldsymbol{a}}_i | \hat{\boldsymbol{a}}_i = \boldsymbol{a}_i, i \in \mathbb{Z}, 1 \leq i \leq n\}$. Here, set $\hat{\mathbf{P}}$ contains all possible positions for each atom i, and set $\hat{\mathbf{A}}$ contains the corresponding atom feature vector for each atom i.

For crystal property prediction tasks, $\mathbf{L}$ representing the minimum repeating patterns is given. In the following proofs, we don't consider the case that the provided $\mathbf{L}' = \boldsymbol{\alpha}\mathbf{L}$ for $\boldsymbol{\alpha} \in \mathbb{N}_+^3$, which means the provided periodic patterns are not the minimum repeating patterns for a given crystal. When $\mathbf{L}$ representing the minimum repeating patterns is given, the periodic invariance lies in the invariance to shifts of periodic boundaries. We prove that the multi-edge graph construction and fully connected graph construction for crystals employed by our Matformer satisfy periodic invariance.

Multi-edge graph. The Multi-edge graph satisfies the periodic invariance by forming edges between node i and j using all items in the set $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}, d_{ij} \leq r\}$, where r is a prefixed threshold. It can be seen that all pairwise Euclidean distances between node i with positions $\{\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$, and node j with positions $\{\hat{\boldsymbol{p}}_j = \boldsymbol{p}_j + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$ are considered, and the shifts of periodic boundaries will not influence the pairwise Euclidean distances. Thus, this method satisfies periodic invariance. Additionally, by using pairwise Euclidean distances, unit cell E(3) invariance is naturally satisfied.

We note that for a given node i, the radius can be computed by a deterministic function that takes all pairwise distances between node i and all other nodes as input, and produces a real value r as output. This will not break periodic invariance due to the fact that the deterministic function will always produce the same r for the same input $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}, 1 \leq j \leq n\}$. One example is that we can use the 12-th smallest distance d_{12} in $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}, 1 \leq j \leq n\}$ as the radius for node i.

Fully connected graph for crystals. To construct fully connected graphs for crystals, in our design, node i represents atom with atom feature $\mathbf{a}_i$ and all its repeats in the infinite 3D space, with positions $\{\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$. In the t-fully-connected graph for crystals, for node i and j, t smallest pairwise distances in $\{d_{ij} | d_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j + k_1^{'}\boldsymbol{\ell}_1 + k_2^{'}\boldsymbol{\ell}_2 + k_3^{'}\boldsymbol{\ell}_3||_2,\ k_1^{'}, k_2^{'}, k_3^{'} \in \mathbb{Z}\}$ are considered, where t is a hyperparameter to control the edge number. Similar to the proof of multi-edge graph, all pairwise Euclidean distances between node i with positions $\{\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$, and node j with positions $\{\hat{\boldsymbol{p}}_j = \boldsymbol{p}_j + k_1\boldsymbol{\ell}_1 + k_2\boldsymbol{\ell}_2 + k_3\boldsymbol{\ell}_3,\ k_1, k_2, k_3 \in \mathbb{Z}\}$ are considered, and the shifts of periodic boundaries will not influence the pairwise Euclidean distances. Hence, this method satisfies periodic invariance. In addition, the usage of pairwise Euclidean distances makes the unit cell E(3) invariance naturally satisfied.

A.3 Dataset descriptions

We show the detailed dataset Descriptions for two crystal datasets used in experimental studies, including The Materials Project and JARVIS.

The Materials Project dataset. In particular, for tasks of formation energy and band gap, we directly follow ALIGNN [3] and use the same training, validation, and test set, including 60000, 5000, and 4239 crystals, respectively. For tasks of Bulk Moduli and Shear Moduli, we follow GATGNN [7], the recent state-of-the-art method for these two tasks, and use the same training, validation, and test sets, including 4664, 393, and 393 crystals. In Shear Moduli, one validation sample is removed because of the negative GPa value. We either directly use the publicly available codes from the authors, or re-implement models based on their official codes and configurations to produce the results. Detailed configurations of these retrained models are provided in Appendix. A.5.

The JARVIS dataset. JARVIS is a newly released database proposed by Choudhary et al. [4]. For JARVIS dataset, we follow ALIGNN [3] and use the same training, validation, and test set. We evaluate our Matformer on five important crystal property tasks, including formation energy, bandgap(OPT), bandgap(MBJ), total energy, and Ehull. The training, validation, and test set contains 44578, 5572, and 5572 crystals for tasks of formation energy, total energy, and bandgap(OPT). The numbers are 44296, 5537, 5537 for Ehull, and 14537, 1817, 1817 for bandgap(MBJ). The used metric is test MAE. The results for CGCNN and CFID are taken from ALIGNN [3], other baseline results are obtained by retrained models. Detailed configurations of these retrained models are provided in Appendix. A.5.

A.4 Matformer configurations

We show the detailed configurations of our Matformer models in this section.

Notations. $\boldsymbol{a}_i \in \mathbf{A}$ is the d_a-dimensional feature vector for node i, $\boldsymbol{e}_{ij}^h \in \mathbf{E}$ is d_e-dimensional feature vector for the h-th edge between nodes i and j, and $\boldsymbol{f}_i^\star$ is the input feature vector of node i for a given layer of Matformer. LN_Q, LN_K, and LN_E denote the linear transformations to compute query, key, and edge embedding in a given Matformer layer, respectively. $\boldsymbol{q}_i$, $\boldsymbol{k}_i$ and $\boldsymbol{v}_i$ are the computed query, key, and value vectors after these linear transformations for node i. $\boldsymbol{e}_{ij}^{h'}$ is the intermediate output in a given Matformer layer for $\boldsymbol{e}_{ij}^h$.

Crystal graph construction. For all Matformer models used in two material datasets, including The Materials Project and JARVIS, we use the radius-based multi-edge graph construction method. We use the 12-th smallest distance between the given atom and all nearby atoms as radius, and include all nearby atoms within the radius as the neighborhood for the given atom. It satisfies periodic invariance as described in Appendix. A.2. For the encoding of periodic patterns, we use six geometric distances, including $||\boldsymbol{\ell}_1||_2$, $||\boldsymbol{\ell}_2||_2$, $||\boldsymbol{\ell}_3||_2$, $||\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2||_2$, $||\boldsymbol{\ell}_2 + \boldsymbol{\ell}_3||_2$, and $||\boldsymbol{\ell}_1 + \boldsymbol{\ell}_3||_2$ in our study. For any of these distances, if it is already in the radius of node i computed by the previous graph construction method, it will not be added as a self-connecting edge of node i in the final graph.

Node and edge embeddings. For each node, we map the atomic number to a 92-dimensional embedding using CGCNN [10] atomic embedding. We then use a linear transformation to map it to a 128-dimensional vector as the input $\boldsymbol{f}_i^*$ to the first Matformer message passing layer. For each edge, we map the Euclidean distance to a 128-dimensional embedding using 128 RBF kernels with centers from 0.0 to 8.0. It is then mapped to a 128-dimensional vector as the edge input $\boldsymbol{e}_{ij}^h$, by a nonlinear layer followed by a linear layer.

Matformer layer. For message passing, we use five layers of Matformer Message Passing layer, with four attention heads. The embedding sizes for $\boldsymbol{q}_i$, $\boldsymbol{k}_i$, $\boldsymbol{v}_i$ for a single head are 128, mapped from $\boldsymbol{f}_i^*$ using corresponding linear transformations LN_Q, LN_K, and LN_V. For a given layer, different heads use different LN_Q, LN_K, LN_V, and LN_E, but share other operations. To obtain the final message m_i, the features from four heads are concatenated and mapped to a 128-dimensional vector by a linear transformation.

Readout layer. We use the mean pooling to aggregate features from all nodes in a graph and then use a nonlinear layer with hidden dimension 128 followed by a linear layer to obtain the scalar output for a crystal graph.

Training hyperparameters. For all tasks in The Materials Project and JARVIS, we use the Adam optimizer [5] with weight decay [6] of 1e-5 and one cycle learning rate scheduler [9]. We use the batch size of 64. We use mean square error as the objective function to train and mean absolute error as the evaluation metric to validate and test. We only slightly adjust the learning rates and training epochs for different tasks in The Materials Project and JARVIS, as shown in Table. 1 and Table. 2, respectively. We use Pytorch to implement our models. For all tasks on two benchmark datasets, we use one NVIDIA RTX A6000 48GB GPU to train our Matformer models.

Table 1: Training hyperparameters for Matformer models on The Materials Project.

Parameter	Formation Energy	Bandgap	Bulk Moduli	Shear Moduli
Learning rate	1e-3	5e-4	1e-3	1e-3
Epoch number	500	500	500	300

Table 2: Training hyperparameters for Matformer models on JARVIS.

Parameter	Formation Energy	Bandgap(OPT)	Total Energy	Ehull	Bandgap(MBJ)
Learning rate	1e-3	8e-4	1e-3	1e-3	1e-3
Epoch number	500	300	500	500	300

Table 3: Comparison in terms of test MAE on The Materials Project dataset. We show results both from retrained models and referred papers to make the comparison clear and fair. Results from original papers are shown in parentheses () on the right. '-' denotes no results are reported in referred papers. The best results are shown in **bold** and the second best results are shown with underlines.

	Formation Energy	Band Gap	Bulk Moduli	Shear Moduli
Method	eV/atom	eV	log(GPa)	log(GPa)
CGCNN [10]	0.031 (0.039)	0.292 (0.388)	0.047 (0.054)	0.077 (0.087)
SchNet [8]	0.033 (0.035)	0.345 (-)	0.066 (-)	0.099 (-)
MEGNET [2]	0.030 (0.028)	0.307 (0.33)	0.060 (0.050)	0.099 (0.079)
GATGNN [7]	0.033 (0.039)	0.280 (0.31)	0.045	0.075
ALIGNN [3]	0.022	0.218	0.051 (-)	0.078 (-)
Matformer	**0.021**	**0.211**	**0.043**	**0.073**

A.5 Configurations of retrained models

We show configurations of retrained models for The Materials Project and JARVIS in this section. The results from their original papers are shown in Table. 3.

SchNet [8]. We use six layers of SchNet message passing layer following the original paper, with feature dimension of 64. We train SchNet on these four tasks with learning rate of 5e-4 and batch size of 64 for 500 epochs. The Adam optimizer is used with 1e-5 weight decay. One cycle learning rate scheduler is also used. We use the 12-th smallest distance between the given atom and all nearby atoms to serve as the radius for this given atom for the tasks of formation energy and bandgap in The Materials Project and all tasks in JARVIS. For the tasks of shear moduli and bulk moduli, We use the 32-th smallest distance between the given atom and all nearby atoms to serve as the radius for every atom to better the performance.

MEGNET [2]. Following the original paper, we use three layers of MEGNET message passing layer with the same feature dimensions as mentioned in the paper, and use Set2Set readout function. We train MEGNET on these four tasks with learning rate of 1e-3 and batch size of 128 for 1000 epochs following the configuration settings mentioned in the original paper. The Adam optimizer is used with 1e-5 weight decay. One cycle learning rate scheduler is also used. We try different hyperparameters of crystal graph construction and batch size to better the performances of MEGNET. In particular, for formation energy and bandgap, we use a radius of 4.0 for all atoms with batch size of 128. And for bulk moduli and shear moduli and all tasks in JARVIS, we use the 12-th smallest distance between the given atom and all nearby atoms to serve as the radius for this given atom to better the performance, and the batch size of 64 is chosen because of better empirical results.

CGCNN [10]. For CGCNN, we directly use the publicly available code from Xie and Grossman [10], with 128 hidden dimensions, batch size of 256, and three layers of CGCNN message passing layer. We train CGCNN with Adam optimizer because of better empirical results. We train CGCNN for these four tasks using learning rate of 1e-2 for 100 epochs and 1e-3 for the next 900 epochs following the official code. The radius cutoff of 8.0 is used for all atoms, and the nearest 12 neighbors are selected.

GATGNN [7]. For GATGNN, we directly use the publicly available code from Louis et al. [7], with 128 hidden dimensions, batch size of 256 and keep other default settings. We train GATGNN with learning rate of 5e-3 with learning rate decay milestone of 300 epochs and decay parameter of 0.5. We train GATGNN for 500 epochs for the task of formation energy and bandgap in The Materials Project and all tasks in JARVIS, with early stop. The radius cutoff of 4.0 is used for all atoms, and the nearest 16 neighbors are selected according to the original paper.

ALIGNN [3]. For ALIGNN, we directly use the publicly available code from Choudhary and DeCost [3]. We use the official best model configurations of ALIGNN to train ALIGNN models on the tasks of bulk moduli and shear moduli, with learning rate of 1e-3 and batch size of 64. In particular, we use ALIGNN with four gcn layers and four alignn layers.

Overall, we either directly use the publicly available codes from corresponding authors [10, 7, 3], or re-implement models based on their official codes and configurations [8, 2] to produce the results in our experiments.

A.6 Building block of Matformer

We provide ablation studies about sigmoid and layernorm instead of softmax in our Matformer message passing layer in this section.

We evaluate the operation of sigmoid and normalization instead of softmax in our message passing, which is designed to capture the node degree information in multi-edge crystal graphs, described in Sec. 4.2. Compared with using softmax with scalar attention coefficient, denoted as Softmax-scalar, or softmax with vector attention coefficient, denoted as Softmax-vector, using sigmoid and normalization leads to significant performance improvement in terms of test MAE, justifying the effectiveness of this operation in Matformer message passing design, as shown in Table. 4.

Table 4: Operation ablation study.

Operation	Test MAE
Softmax-scalar	0.0376
Softmax-vector	0.0347
Sigmoid and Norm	**0.0325**

A.7 Complexity analysis of introducing angular information

We provide the complexity analysis of adding angular information to Matformer in this section.

Assume that we have n atoms in a single cell and thus n nodes in the original multi-edge graph. Also assume that every node has at least 12 neighbors following the graph construction method of Matformer and assume there are no self-connecting edges. This will result in a graph $G = (V, E)$ where $|V| = n$ and $|E| = 12/2n = 6n$. When converting graph G into line graph $L(G)$, every edge is treated as a node in the line graph. So we have $6n$ nodes in the line graph. Every edge in the original graph is connecting 2 nodes and every node has $(12 - 1)$ other edges, resulting in 22 neighboring edges for each edge in the original graph. So we have $22 * 6n/2 = 66n$ edges in the converted line graph. Compared with original graph with $|V| = n$ and $|E| = 6n$, the converted line graph is super large with node number of $6n$ and edge number of $66n$.

References

[1] Lowik Chanussot*, Abhishek Das*, Siddharth Goyal*, Thibaut Lavril*, Muhammed Shuaibi*, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open Catalyst 2020 (OC20) Dataset and Community Challenges. ACS Catalysis, 2021. doi: 10.1021/acscatal.0c04525.

[2] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals. Chemistry of Materials, 31(9):3564–3572, 2019.

[3] Kamal Choudhary and Brian DeCost. Atomistic Line Graph Neural Network for improved materials property predictions. npj Computational Materials, 7(1):1–8, 2021.

[4] Kamal Choudhary, Kevin F Garrity, Andrew CE Reid, Brian DeCost, Adam J Biacchi, Angela R Hight Walker, Zachary Trautt, Jason Hattrick-Simpers, A Gilad Kusne, Andrea Centrone, et al. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. npj Computational Materials, 6(1):1–13, 2020.

[5] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, 2015.

[6] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In International Conference on Learning Representations, 2018.

[7] Steph-Yves Louis, Yong Zhao, Alireza Nasiri, Xiran Wang, Yuqi Song, Fei Liu, and Jianjun Hu. Graph convolutional neural networks with global attention for improved materials property prediction. Physical Chemistry Chemical Physics, 22(32):18141–18148, 2020.

[8] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A Continuous-filter Convolutional Neural Network for Modeling Quantum Interactions. Advances in Neural Information Processing Systems, 30, 2017.

[9] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. 2018.

[10] Tian Xie and Jeffrey C Grossman. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. Physical review letters, 120(14): 145301, 2018.

[11] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Badly for Graph Representation? Advances in Neural Information Processing Systems, 34, 2021.